**合约地址：**https://etherscan.io/address/0xc2438ef1d2f2c1427bf0a39f02dba6a4a9de4793#code

**漏洞说明：** BankDeposit为运行在以太坊主链上的智能合约，其中函数init（第23-28行）将Owner的权限转给了调用者（msg.sender），该函数原本为一个构造函数，其函数名需要与合约名相同，但是开发者错误地将构造函数名写错，导致init变为一个普通的公开函数，从而导致权限的丢失。

**合约部分代码如下：**

7: contract BankDeposit {

12: address Owner;

18: mapping (address => uint) public Deposits;

23: function init() payable open {

24: Owner = msg.sender;

28: }

50: function withdrawTo(address to, uint amount) public onlyOwner {

51: if (WithdrawEnabled()) {

52: uint max = Deposits[msg.sender];

53: if (max > 0 && amount <= max) {

54: to.transfer(amount);

56: }

57: }

58: }

63: } }

**攻击步骤：**

步骤一：攻击者可以先调用init函数将Owner变量改为自己的地址（第24行）

步骤二：攻击者在修改Owner后有权限调用withdrawTo函数（第50-58行），将调用参数amount设置为小于等于自己账户deposit的值，转走amount数量的以太币（第54行）

步骤三：攻击者重复步骤二，多次调用withdrawTo函数并最终转走多于自己账户deposit数量的以太币完成攻击。

**攻击演示：**

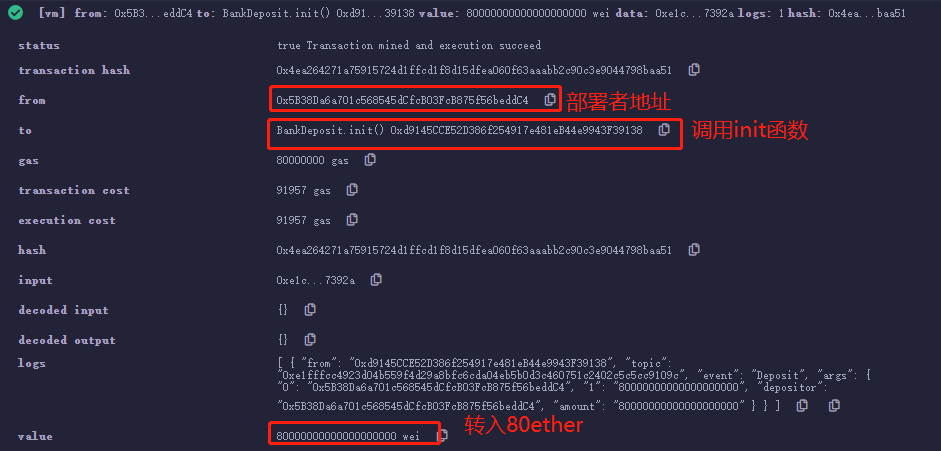
我们通过Remix平台复现针对BankDeposit合约的攻击过程。

* 部署合约账户地址：0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
* 合约地址：0xd9145CCE52D386f254917e481eB44e9943F39138
* 攻击者地址：0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

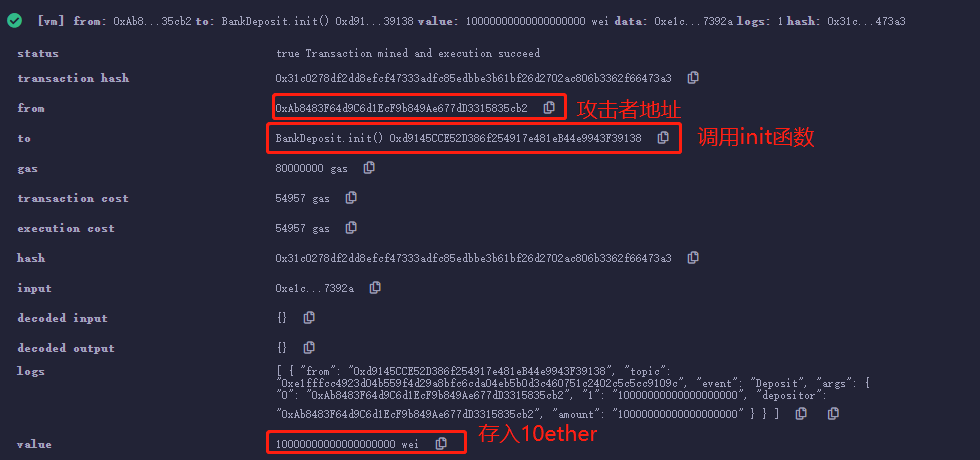
**环境准备: (部署者操作(0x5B38Da6a701c568545dCfcB03FcB875f56beddC4))**

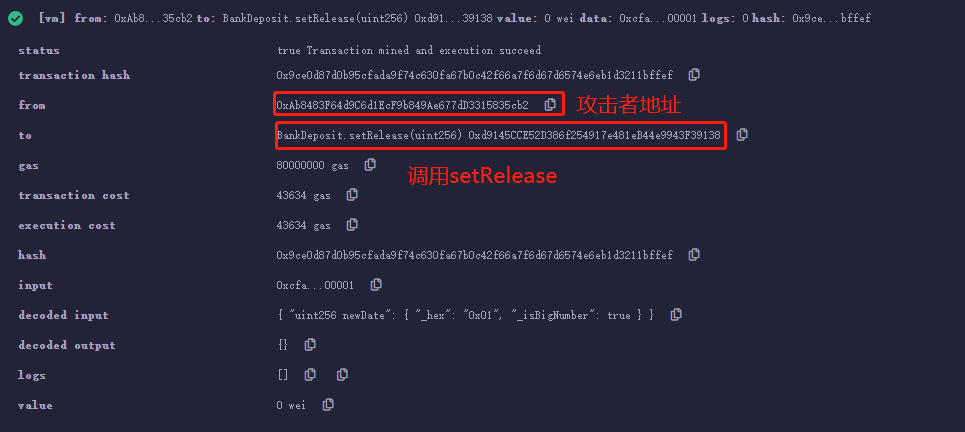
步骤一：部署合约



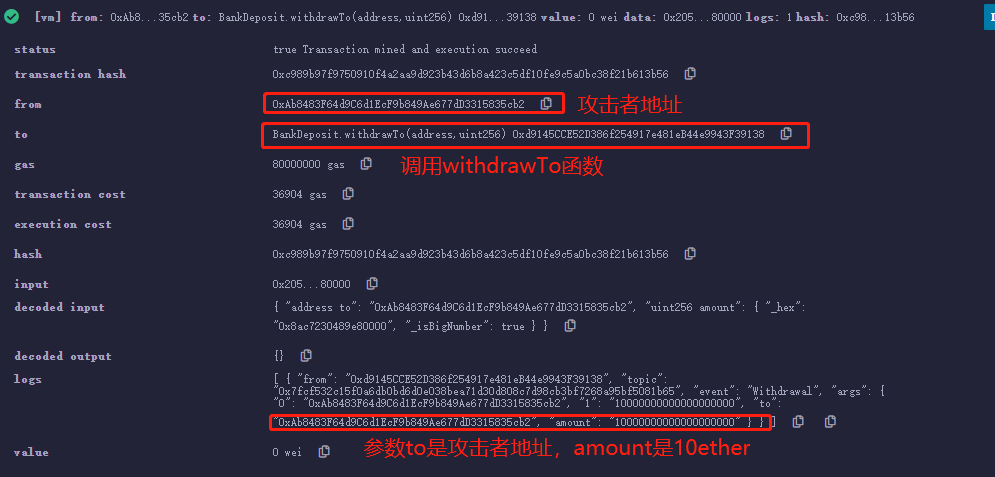
步骤二：调用init()函数，并给合约转入80ether

**攻击过程: (攻击者操作(0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2))**

步骤一：调用init()函数获得Owner权限，并将10ether存入自己的deposit

步骤二：调用setRelease()函数设置Date

步骤三：调用withdrawTo()函数转出10ether

步骤四：再次调用withdrawTo()函数转出10ether

**攻击结果：**

攻击者成功得到多于自己存入数量的ether。